Motivation and Objective

Motivation and Objective

It is often the case that images are taken with imperfections and that the main object of focus is captured with some level of blur. Furthermore, it is sometimes important and necessary to reclaim information from these kinds of images. A classic example is a still photograph which captures a car speeding linearly across it (i.e. left to right). In criminal investigations, it may be necessary to deblur such an image in order to obtain information about the vehicle's physical characteristics. Such details may include unusual markings, license plate number, etc. which can be of use as evidence in the investigation. Thus, our endeavor of deblurring is a useful and worthwhile mission to undertake.

Our objective is to explore the usage of the fluttered shutter in deblurring linear, one-dimensional motion blur and its effectiveness of deblurring in the time domain compared to in the frequency domain. We also aim to compare the effectiveness of this method with that of the traditional boxcar-filter deblurring when different levels of noise are present.

Abstract

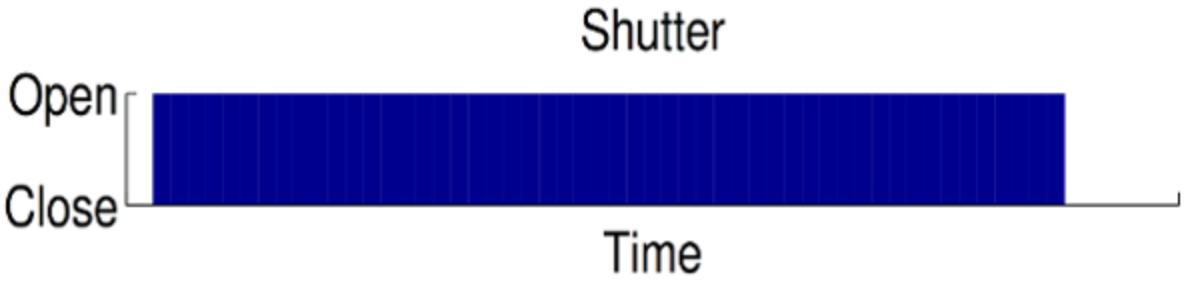
Abstract

We would like to simulate and explore the deblurring of images with linear, one-dimensional blur. This blur is the result of the object being in 1-D motion while the shutter is open. The effect is similar to convolving a boxcar filter and smearing the image in it's direction of moving. The boxcar filter has a “sinc” spectrum, which causes the high-frequency details to be lost. In addition, deconvolving a boxcar also magnifies the high frequency noise. These together render the result of recovery undesirable.

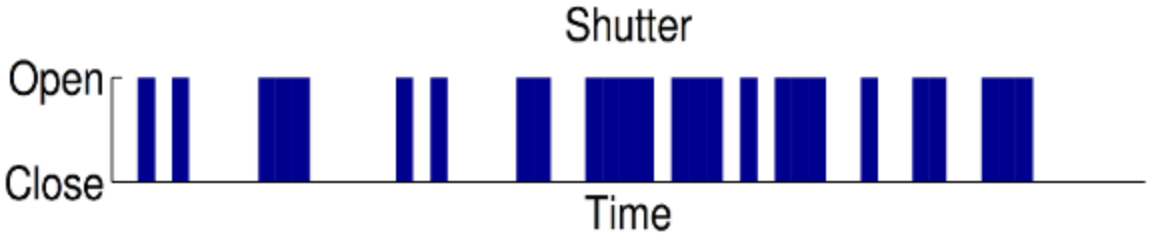
By fluttering the shutter during the exposure with a random binary sequence, we broaden the smearing filter spectrum, which preserves more high frequency details in the blurred image. This produces a decent deconvolution result with less noise. We also exploited and compared the result of deblurring in both time and frequency domain using both coded and uncoded shutter.

Why Flutter Shutter

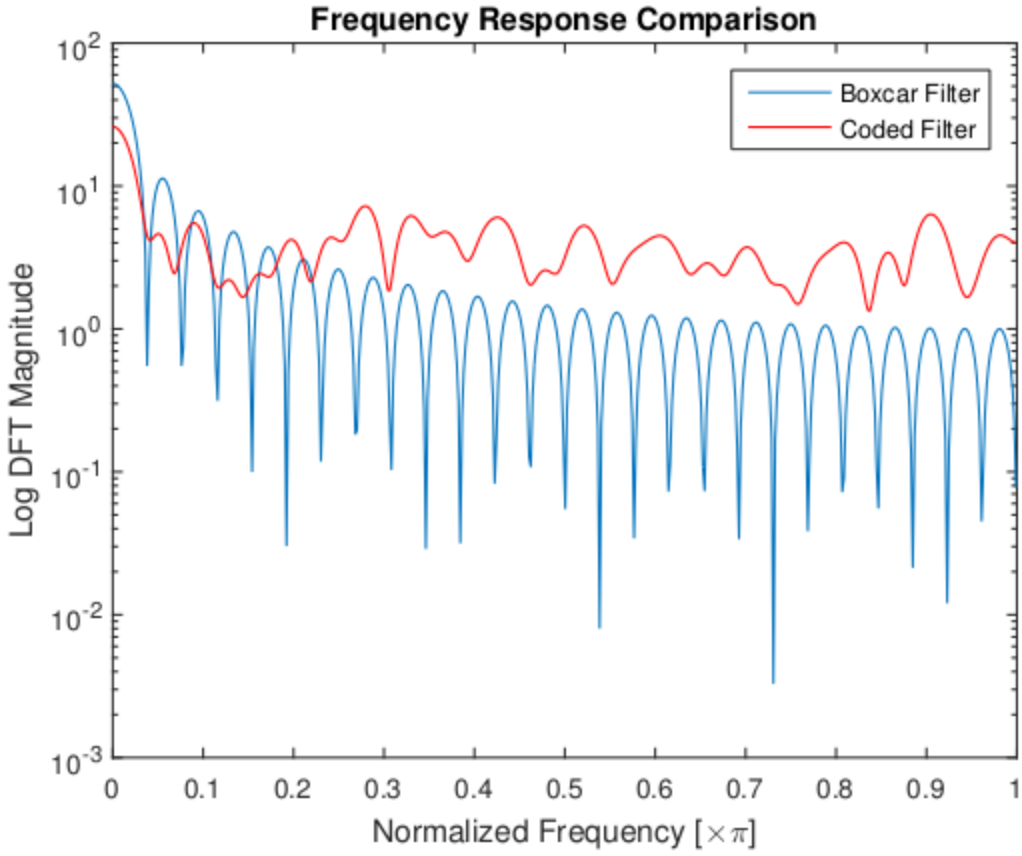
Why Fluttered Shutter?



The fourier transform of the boxcar filter, as shown in blue lines in the spectrum below, is a sinc function. The sinc function has zeros at certain frequencies, and thus the information contained in those frequencies are lost entirely when the blurred image is generated. Also, the envelope of the sinc function decays rather quickly at high frequencies. We know that the noise has constant magnitude at all frequencies. So when we try to deblur the image by deconvolution, the noise at high frequencies will be greatly amplified.



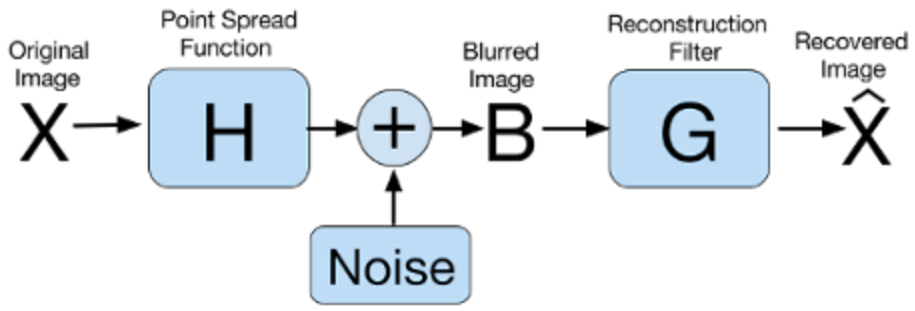
By using a fluttered shutter, which opens and closes during camera exposure according to a random binary sequence (a “1” corresponds to “open”, “0” to “close”), the fluttered shutter changes the boxcar filter to a broad-band filter that preserves high frequency spatial details, as shown in the above image in red-colored lines. Also, the flutter shutter filter has no zeros in its frequency response. This allows direct deconvolution to be possible without immediate destruction of image quality.


Frequency Response Comparison
Boxcar Filter
Coded Filter
Log DFT Magnitude
Normalized Frequency [×π]
10^2
10^1
10^0
10^-1
10^-2
10^-3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Summary of Approach

Summary of approach

Our approach includes three parts: 1. generation of blurred images, 2. deblur, 3. evaluation.



To generate the blurred images, we first convolve the impulse response of the boxcar and coded filter with the original image. Then, we add zero mean Gaussian noise to the blurred images to simulate the thermal and readout noise caused by real-world camera sensors.

We implemented the deblurring process for both the traditional shutter (boxcar filter) and the flutter shutter (coded filter) in both time and frequency domains. In time domain implementation, we use least square estimation to solve the linear equation $B = AX + n$ for X. In the above equation, B is the blurred image, A is the smear matrix corresponding to the filter type, and n is noise. In the frequency domain implementation, we divide the DFT of the blurred image by the DFT of the filter response, and take the inverse DFT to get our deblurred image.

After getting the deblurred images, we calculated the PSNR (explained in Results Section) to characterize how well each approach works. We plotted the PSNR with respect to the variance of the Gaussian noise we added to the blurred image to determine the deblurring performance of the traditional shutter (boxcar filter) and the flutter shutter (coded filter).

Blurred Image Generation

Blurred Image Generation

We use linear convolution to simulate our test image. For coded blurring, we convolve our test picture with a 52-length coded pulse train $[10100001110....]$, which generates an image resembling one that is taken by a camera with a fluttered shutter. For uncoded blurring, we convolve our test picture with a 52-length boxcar signal, which results in the effect of linear, one-dimensional blur of the entire image. Gaussian white noise is added to the blurred image to mimic the readout noise produced by camera sensors in the real world.

The convolution is done by superpositioning the frames according to the code sequence. This hardcoding approach exactly models the generation of one-dimensional motion blurred images by a real camera. For each value a_j in the code sequence, regardless of 0 or 1, we shift the corresponding frame down by j elements. If $a_j = 1$, we add the corresponding frame to the final blurred image. If $a_j = 0$, then we ignore the corresponding frame.

1. Begin with the original image as the base frame.
2. Look at the code sequence. If the j-th value in the sequence is a `1', then produce another frame that is the base frame shifted downwards by j elements.
3. If k is the number of `1's in the sequence, then we should have k total shifted frames, including the base frame, after the code sequence is entirely processed.
4. We superposition all the shifted frames together to generate the blurred image.
5. We scale the image by k to set pixel values back to the valid interval [0, 1] to avoid over-exposure.
6. We add independent and identically distributed zero-mean Gaussian white noise to each element of the blurred image. The variance of this noise will be our independent/manipulated variable, which characterizes the noise power.
7. We repeat the generation of noisy blurred image for different variances of Gaussian noise.

Boxcar-blur Image

Coded-blur Image

Image Deblur

Image Deblur

Image Resizing

Before implement the deblurring, we resize the blurred image B by mk to compensate for the difference between blur size k and code length m. In the resized image, the blur size is m instead of k.

Frequency Domain Deblur

- **Coded:** We obtain the frequency response of the filter H(k) and the frequency content of the blurred image B(k) through DFT. The system can be represented in frequency domain as $B_j(k) = H(k)X_j(k) + N(k)$ where $B_j(k)$ and $X_j(k)$ are the DFT of the j-th column of the blurred image and the original image, respectively. $N(k)$ represents the noise. We divide $B_j(k)$ by $H(k)$ to obtain $X_j(k)$. We then took the inverse DFT of $X(k)$ to obtain the recovered image.

- **Boxcar:** We use almost the same frequency domain deblur approach as used for the coded filter. The primary difference between the two approaches is the following: the frequency response of a boxcar filter is a sinc function, which contains zeros at certain points. We added a small number to the zeros to avoid division-by-zero problems. So $H(k)$ in this case is a slightly modified version of the $H(k)$ used for the coded filter.

Time Domain Deblur

- **Coded:** We first generate a circulant matrix whose characteristic vector contains the code sequence. Then we kick out the wrap-around part of the circulant matrix and take only the left n columns, which construct the foreground smear matrix Af whose length matches that of the blurred image B. Then we concatenate Af with the background contribution vectors to generate the matrix A for the linear system $B = AX$, where B is the blurred image and X is the original image.

Then we use least square estimation to find the solution for this system, which gives the deblurred image X, the estimate of X. Background contribution vectors are $bk = abs(1 - Af * ones(n, 1))$, where n is the length of the original image. The first m elements in bk account for the background contribution at the top of the blurred image, and the last m elements account for that at the bottom. Thus we generate two vectors of length $n + m - 1$. The first m elements of the first vector are the first m elements in bk, and the last m elements of the second vector are the last m elements in bk. The rest elements of the two vectors are all zeros. These two vectors are then concatenated to Af to generate the smear matrix A.

- **Boxcar:** We use basically the same approach as used in time domain deblurring, except that we use the 52-length boxcar as the characteristic vector of the matrix Af.

Over-exposure Compensation:

The deblurred image might have some element values above 1 or below 0. We set all the values below 0 to 0 and all above 1 to 1.

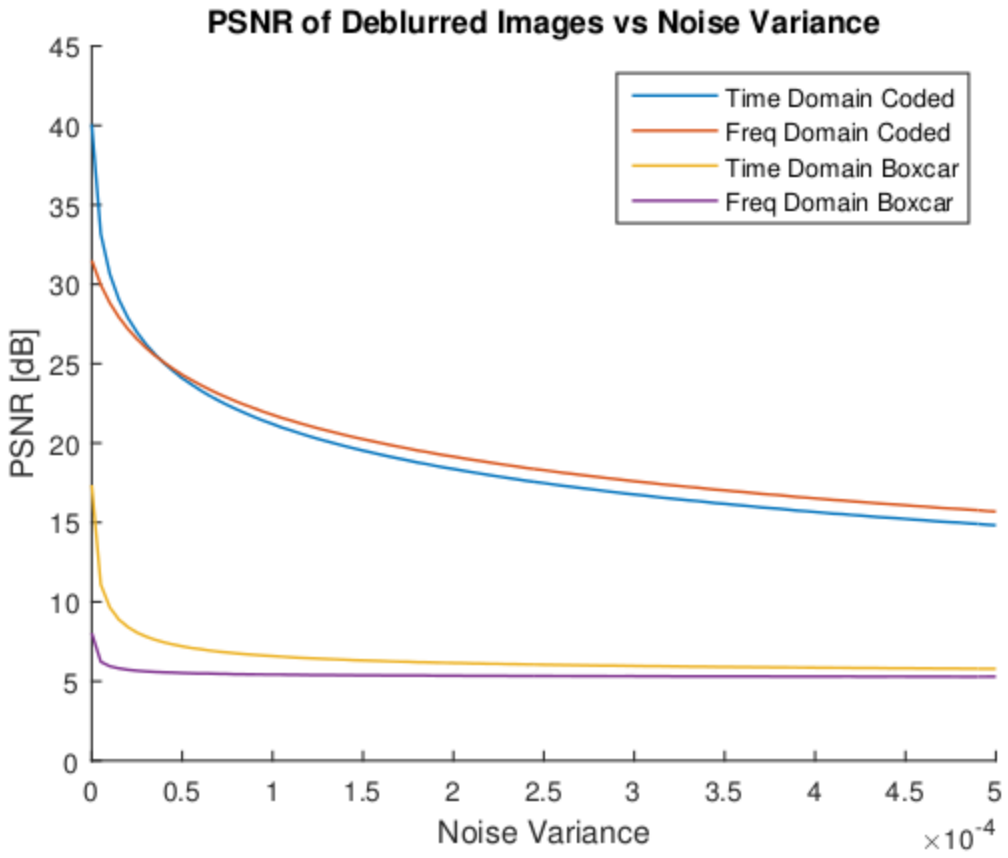
Noise Analysis

Noise Analysis

Normally, all element values of an image matrix fall within the interval $[0, 1]$. However, the introduced noise reduces the deblurring quality, resulting in some elements having values above 1 or below 0 in the deblurred image matrix. This effect has been compensated in the previous part. We then compute the PSNR of the deblurred images with respect to the original/ground-truth and plot the PSNR values against different levels of noise, characterized by the variances of the independent and identically distributed zero-mean Gaussian random variables. The formula of PSNR is $PSNR = 10 \cdot \log_{10}\left(\frac{\mathrm{MA}{X_I}^2}{\mathrm{MSE}}\right) = 20 \cdot \log_{10}\left(\frac{\mathrm{MA}X_I}{\mathrm{MS}E^{0.5}}\right)$ where $\mathrm{MA}X_I$ is the maximum possible pixel value of the image and MSE is the mean squared error between the deblurred image and the ground-truth image.

Results and Conclusion

Results and Conclusion



In the noise analysis, we compute the PSNR as a function of noise variance for a variance range from 0 to 0.0005. We do this for the deblur results of both the flutter shutter blurred image as well as the boxcar filter blurred image. We plot these results, which you can see in figure below. Here, the PSNR characterizes the deblur result quality and the noise variance characterizes noise level. Two important results and conclusions can be drawn here.

Flutter Shutter: The flutter shutter deblur results always have a higher PSNR than the boxcar filter deblur results at all noise variances. This shows that images taken with a flutter shutter camera can be deblurred more successfully than images taken with a traditional boxcar filter in all cases of noise.

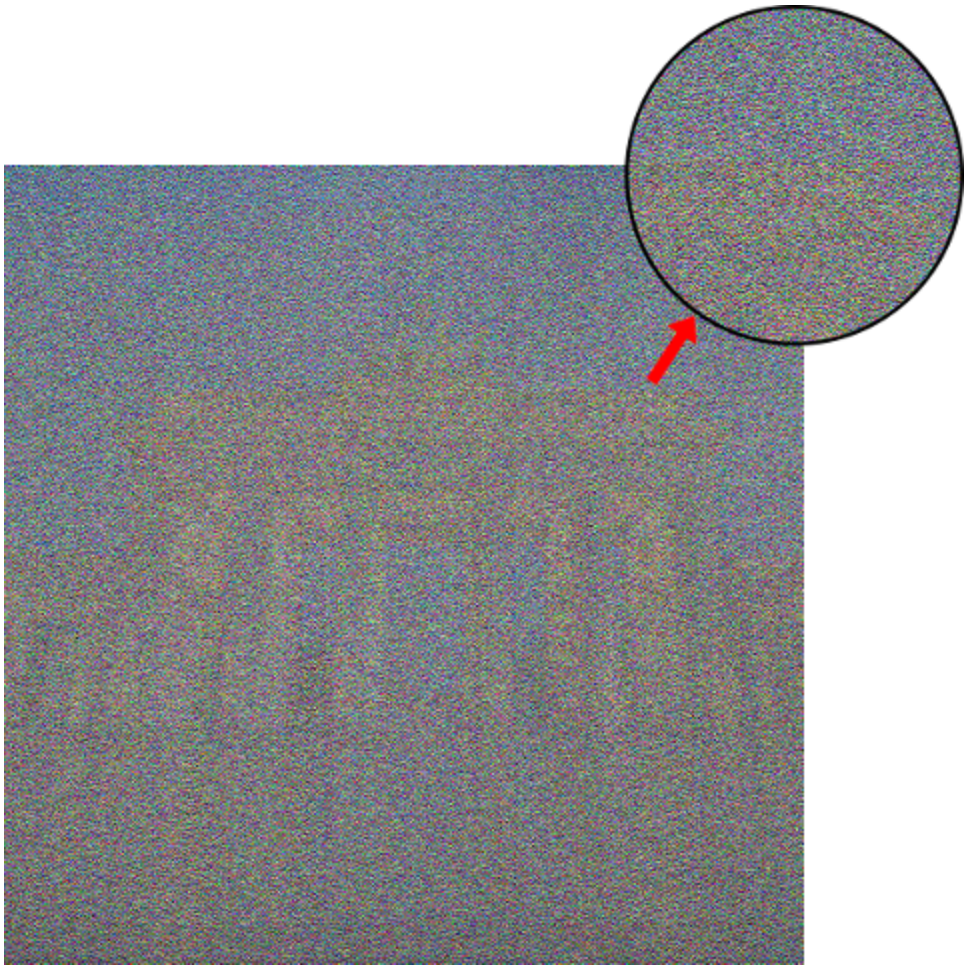
Flutter Shutter Deblur - Noise Variance 0

Flutter Shutter Deblur - Noise Variance 0.0005

Boxcar: The boxcar filter deblur result's PSNR decays extremely quickly to a minimum value of 5 even with only a small amount of noise in the boxcar filter blurred image. On the other hand, the flutter shutter deblur result's PSNR decays rather gradually with increasing noise in the flutter shutter blurred image. This result shows that flutter shutter images and their deblur results are more resistant to noise damage than are boxcar filter images. Deblurred pictures demonstrate this conclusion, showing that even at a noise variance of only 0.0005, the boxcar filter deblur result is already heavily damaged while the flutter shutter deblur result quality is maintained comparatively well. .

Boxcar Deblur Noise Variance 0

Boxcar Deblur Noise Variance 0.0005



Implications and Future Works

Implications

Our result demonstrates that the fluttered shutter improves deblurred image quality significantly. Through a least squares estimation approach, we can avoid the problem of having zeros in the filter spectrum, an issue that was prevalent with the uncoded/boxcar filter.

Future Works

- Exploit deblurring where blur size is larger than object length
- Noise analysis when the signal power is low (i.e. image is less exposed). Focus on low exposure part of the deblurred image.
- Hardware implementation using physical fluttered shutter and camera

References

References

Raskar, Ramesh, Amit Agrawal, and Jack Tumblin. "Coded exposure photography: motion deblurring using fluttered shutter." ACM Transactions on Graphics (TOG) 25.3 (2006): 795-804.

Team Members

Shuqing Chen , Jiaqi Liu , Yulan Shih , Yiqiu Wang

Special Thanks

Vivek Boominathan, Professor Richard Baraniuk

Codes and Poster

Codes

Github: https://github.com/wangyiqiu/imagedeblurelec301.git